ERB-975

NRCC No. 23952

National Research Council Canada
Conseil national de recherches Canada

Division of Electrical Engineering
Division de génie électrique

EFFICIENT SOLUTION FOR THE KINEMATIC POSITIONS FOR THE PUMA 560 ROBOT

S. ELGAZZAR

DECEMBER 1984

Canada

85-3212/

Copyright 1984 by
National Research Council of Canada

Permission is granted to quote short excerpts and to reproduce figures and tables from this report, provided that the source of such material is fully acknowledged.

Additional copies are available free of charge from:

EFFICIENT SOLUTION FOR THE KINEMATIC POSITIONS FOR THE PUMA 560 ROBOT

S. Elgazzar

ERB-973

NRCC No. 23952

DECEMBER 1984

TABLE OF CONTENTS

		Page
ABSTRACT/RÉSUMÉ		v
CHAPTER 1.	INTRODUCTION	1
CHAPTER 2.	DEFINITIONS	2
2.1	Robot Joint Variables and Cartesian Vector	3
2.2	Arm Configuration	6
2.3	Singular Points	7
CHAPTER 3.	DIRECT KINEMATIC SOLUTION	8
3.1	Cartesian Position/Orientation Vector	8
3.2	Arm Configuration Parameters	12
CHAPTER 4.	INVERSE KINEMATIC SOLUTION	13
4.1	Computation of θ_1	13
4.2	Computation of θ_2 and θ_3	15
4.3	Computation of θ_4, θ_5, and θ_6	17
CHAPTER 5.	SUMMARY AND DISCUSSION	18
REFERENCES		19
APPENDIX A.	PUMA 560 PARAMETERS	21
A1.	LINK CONSTANTS	21
A2.	TRANSFORMATIONS	21
A3.	JOINT LIMITS	22
APPENDIX B.	PROGRAM LISTINGS	23

LIST OF FIGURES

1	Robot arm: degrees of joint rotation and member identification	3
2	(a) Robot arm: initial state (b) Representation of link ℓ_3	4
3	Link representation for the PUMA 560 Arm	6
4	Error in wrist position due to a wrong reference for θ_3 measurement	11
5	(a) Projection of the first three links on the x, y plane, (b) calculation of θ_1, for left arm, (c) calculation of θ_1, for a right arm	14
6	Calculation of θ_2 and θ_3 for a left arm	15
A1	Range on joint angle θ_6	22

LIST OF TABLES

		Page
1	Conditions on k_1 and θ_3 for elbow configuration	13
2	Arm configuration for the first three links and the sign of the angles involved	16
3	Computation complexity for the kinematics solution of the PUMA 560 arm	18

ABSTRACT

An efficient solution for the kinematic positions for the six degree-of-freedom PUMA 560 robot is presented. Kinematic positions are defined as the transformation from Cartesian coordinates to joint angles and vice-versa. This method is based on the Featherstone method where the special geometry of the robot is fully exploited in the derivation of the solution. Special attention is given to the arm configuration in both directions of the transformation.

The correspondence between the joint angles and position vector as defined in this report and the joint angles and position vector as defined for the PUMA is described in Appendix A as well as the transformation between the Euler angles and the PUMA O, A, T angles. A listing of all the C-programs used in this study can be found in Appendix B. Limit checks on the joint angles are briefly discussed in Appendix A and included in the programs.

RÉSUMÉ

Une méthode de résolution efficace des positions cinématiques du robot PUMA 560 à six degrés de liberté est presentée. Les positions cinématiques sont définies comme la transformation des coordonnées cartésiennes en angles de joint et vice-versa. Cette méthode est fondée sur la méthode Featherstone dans laquelle la géométrie spéciale du robot est entièrement exploitée dans l'établissement de la solution. Une attention spéciale est accordée à la configuration du bras dans les deux sens de la transformation.

La correspondance existant d'une part entre les angles de joint et le vecteur de position définis dans le présent document et d'autre part les angles de joint et le vecteur de position définis pour le PUMA est précisée à l'Annexe A, de même que la transformation entre les angles d'Euler et les angles PUMA (O, A, T). L'Annexe B donne la liste de tous les programmes ayant servi à l'étude. Les contrôles de limite des angles de joint sont brièvement débattus à l'Annexe A et sont inclus dans les programmes.

EFFICIENT SOLUTION FOR THE KINEMATIC POSITIONS FOR THE PUMA-560 ROBOT

S. Elgazzar

CHAPTER 1. INTRODUCTION

The kinematic analysis of a computer-controlled manipulator consisting of six degrees of freedom is considered in this report.

The kinematic problem deals with the analytical study of the spatial configuration of the robot arm as a function of time, in particular the relations between the joint variable space and the Cartesian space. The transformation between the two spaces is very important since robots are usually servoed in the joint space whereas tasks are defined and objects manipulated in the Cartesian space.

The kinematic problem consists of two sub-problems:

(i) the direct kinematic problem in which the joint variables are given and the position and orientation vector of the end effector is found

(ii) the inverse kinematic problem in which the position and orientation vector of the end effector is given and the joint variables should be found

If the desired motion is not known in advance, these computations must be performed in real time so as to obtain a smooth, regular motion of the arm.

General methods exist for the solution of the direct kinematic problem that are efficient and numerically well behaved (1-4). The solution of the inverse kinematic problem is difficult for general geometry and closed-form solution generally does not exist. There are a number of special arrangements of the manipulator joint axes for which closed-form solution exists. Manipulators are usually designed to satisfy two simplifying kinematic criteria:

(i) neighbouring joint axes are oriented at 0° or 90° relative to each other

(ii) the three joints nearest the end effector have their axes intersecting at a point, which is kinematically equivalent to a spherical wrist

The spherical wrist allows the decomposition of the six degree-of-freedom inverse kinematic problem into two three degree-of-freedom kinematic sub-problems. The first of these deals with the arm motion and consists of finding the values of the first three joint variables so as to position the wrist correctly. The second deals with the wrist motion and consists of finding the values of the wrist joint angles that orient the end effector correctly.

Reasonably efficient algorithms have been developed for these transformations (1, 2, 4). The Featherstone method (1) seems to be the most efficient method, especially if one is interested in solving for the inverse kinematic velocities and accelerations. The velocity transformation (1) requires the intermediate and final results of the position transformation to be available and, since almost always this transformation is needed, the velocity transformation is very efficient. The same can be said about the acceleration that requires the intermediate and final results of the position and velocity transformations.

The approach used by Featherstone is applied here to the PUMA 560 robot. The Featherstone algorithm is specific to manipulators having co-planar links with six revolute joints and a spherical wrist. The PUMA robot is different from a co-planar robot by having offsets at the shoulder and elbow joints and also having an asymmetry in the forearm link that shows in the vertical plane. These differences will affect the solution of the first three joints.

This report presents a closed-form solution to perform direct and inverse kinematic positions for the PUMA 560 robot. In addition, special attention is given to the arm configuration. Parameters are defined and used to find the correct solution for a specific configuration when solving for the inverse kinematics. A method to obtain the arm configuration parameters from a known set of joint angles is presented.

Limit checks on the joint angles when solving for the inverse kinematics and a brief discussion of the singularities are also included. The equivalence between the different representation of orientation, PUMA's O, A, T angles, Euler α_e, β_e, ψ_e angles, and this report's r_ρ, r_θ, r_ψ angles, is presented in Appendix A. The listing of the C-programs is given in Appendix B.

CHAPTER 2. DEFINITIONS

In this chapter a description of the PUMA 560 robot is presented. The reference state of the arm and the arm configuration parameters are defined. A brief discussion of the singularities is also given.

2.1 Robot Joint Variables and Cartesian Vector

The PUMA 560 robot has six revolute joints arranged as shown in Fig. 1.

Joint 1 axis coincides with the centerline of the "trunk", link ℓ_1, which coincides with the z-axis of the base coordinate system. The joint angle θ_1 is measured in the counterclockwise direction from the positive y-axis.

The joint 2 axis is perpendicular to and intersects the joint 1 axis and coincides with the centerline of the "shoulder". The "shoulder" is the offset of length d_1 between the "trunk" and the "upper arm". This offset is parallel to the x-y plane and is in the negative x-axis direction when θ_1 is equal to zero. Link ℓ_2, the "upper arm", rotates around joint 2 axis an angle θ_2. The angle θ_2 is equal to zero when link ℓ_2 is parallel to the z-axis, as shown in Fig. 2a.

Joint 3, the "elbow", has its axis parallel to the joint 2 axis. The "forearm", the third link, is formed of a two-part link labelled a and b in Fig. 2b. Link ℓ_3 is defined as the vector sum of a and b, that is, the

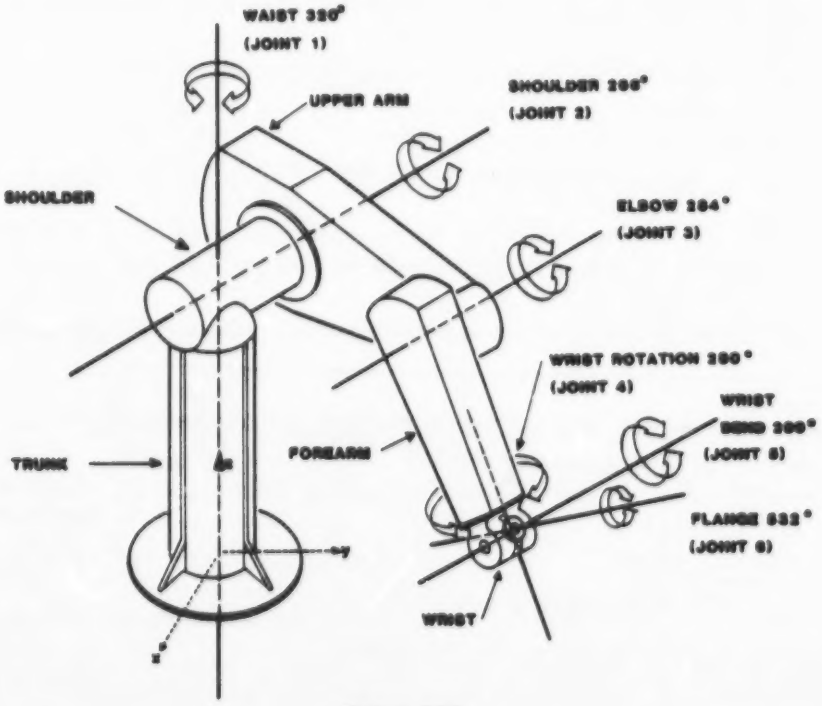
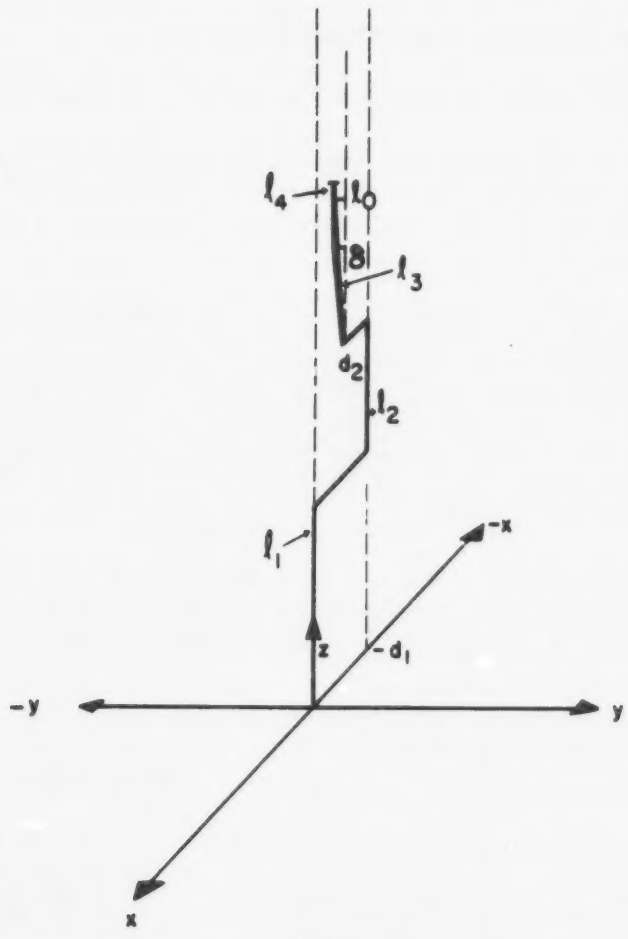
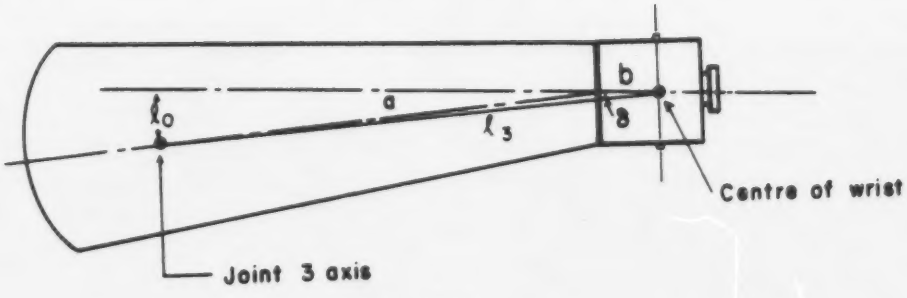


Figure 1. Robot arm: degrees of joint rotation and member identification



(a)



(b)

Figure 2. (a) Robot arm: initial state
(b) Representation of link ℓ_3

distance between the joint 3 axis and the center of the spherical wrist. Link ℓ_3 is offset from link ℓ_2 by a distance d_2 parallel to the x-y plane and rotates an angle θ_3 with respect to link ℓ_2. When θ_1 is equal to zero, d_2 is in or parallel to the positive x-direction. At the reference state, that is when the arm is pointing up (Fig. 2a), link ℓ_2 is parallel to the z-axis and link ℓ_3 makes a known angle δ with the vertical. This angle δ is a function of the arm dimensions. Thus, at the reference position

[2.1] $\theta_{3r} = -\delta$

where

[2.2] $\delta = \sin^{-1} (\ell_o/\ell_3)$

"ℓ_o" is defined as the offset between the centerline of the two parts forming the "forearm", link ℓ_3.

Joints 4, 5, and 6 form a spherical wrist.

Joint 4 axis is perpendicular to and intersects joint 5 axis. Link ℓ_4 is the link from the center of the wrist to the flange or to the end effector if connected. Joint 4 rotates an angle θ_4 around its axis.

The joint 5 axis is parallel to the joints 2 and 3 axes. The angle θ_5 is the angle of rotation of link ℓ_4 measured with respect to the z-axis in the coordinates of link 4 (i.e., rotating the base coordinates through θ_1 $\underline{k}$, then through $-(\theta_2 + \theta_3)$ $\underline{i}$ and finally through θ_4 $\underline{k}$).

The joint 6 axis is perpendicular to and intersects the joint 5 axis. It coincides with the centerline of the gripper mounting flange.

The intersection of the latter three axes in a point is shown in Fig. 1.

The link representation of the robot arm at any arbitrary position is given in Fig. 3. The position of the flange (or alternately the end effector or tool) is expressed in joint coordinates as

[2.3] $\underline{\theta} = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$

and in Cartesian coordinates as

[2.4] $\underline{R} = (r_x, r_y, r_z, r_\rho, r_\theta, r_\psi)$

where $\underline{r} = (r_x, r_y, r_z)$ is the position vector and r_ρ, r_θ, r_ψ are the rotations about the z axis, the new -x axis, and the new z axis that aligns

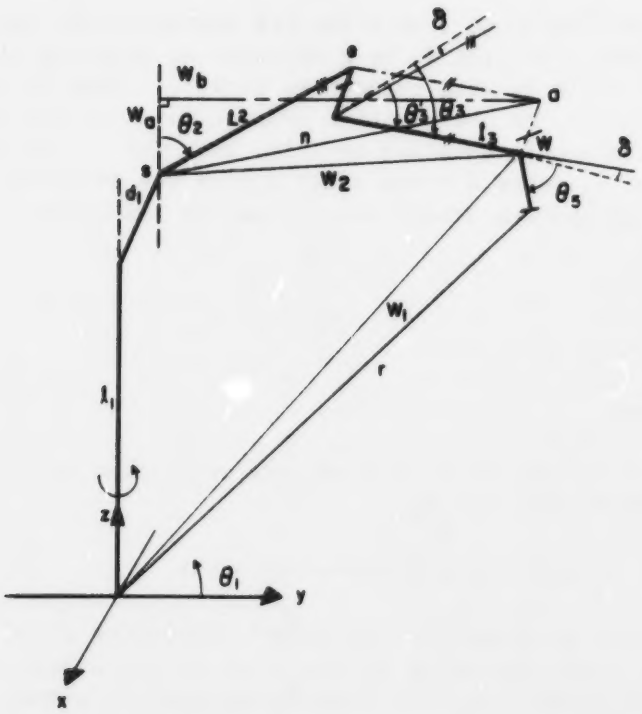


Figure 3. Link representation for the PUMA 560 Arm

the base coordinates with the tip coordinates. The rotation parameters were chosen because they correspond with the arrangement of the joints at the wrist and hence simplify the solution.

2.2 Arm Configuration

The robot arm has similarities with the human arm geometry. Robot manipulators are defined accordingly as having a shoulder, an elbow, and a wrist; see Fig. 1. The robot arm may be righty or lefty, that is, the first three joints of the robot resemble a human's right or left arm, respectively. The elbow can have two configurations; elbow up, where the elbow's position is above the line joining the shoulder to the wrist; and elbow down, where the elbow's position is below that line. Also, the wrist can assume two solutions: the no-flip wrist for which the joint 5 angle is positive and the flip wrist for which the joint 5 angle is negative.

The arm configuration parameters k_1, k_2, and k_3 are defined as follows:

k_1 =	+1	lefty
	-1	righty
k_2 =	+1	elbow up
	-1	elbow down
k_3 =	+1	no-flip
	-1	flip

From the above, it is clear that there are, in general, eight sets of joint angles for each attainable position and orientation. In practice, however, restrictions on the ranges of joint motions reduce the number of solutions. If no solution is found, then the position is outside the manipulator reach.

The parameters k_1, k_2, and k_3 are used to find the inverse kinematic solution and should be specified by the user. However, in the case of a known set of joint angles, as in the case of the direct kinematics, these parameters can be computed, and consequently the arm configuration known, using simple equations derived in the next chapter.

2.3 Singular Points

As shown above, there are eight different solutions to the inverse kinematics depending on the arm configuration. Singular points (also called degenerate points or dead points) are the dividing boundaries between the solution sets. They are also the points where the determinant of the Jacobian is zero and, as a result, there are several solutions for the end effector position.

There are two important properties of the singular points in connection with the control of the hand:

(i) A loss in the number of degrees of freedom occurs. The kinematic equations become less accurate in the neighbourhood of a singular point and totally break down at the point.

(ii) The singular points are the changing points of the kind of the configuration of the arm.

Further discussion of singularities is given in reference 5.

Featherstone (1) provides alternative equations that are accurate in a small neighbourhood around singular points. In this report, these

equations are modified to satisfy the PUMA 560 geometry and the results show good accuracy near the dead points.

CHAPTER 3. DIRECT KINEMATIC SOLUTION

In this chapter, the direct kinematic solution of the PUMA 560 robot is considered. The problem is defined as follows: given the joint angles vector $\underline{\theta} = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$, find the Cartesian position/orientation vector $\underline{R} = (r_x, r_y, r_z, r_\rho, r_\theta, r_\psi)$, and the arm configuration parameters k_1, k_2, and k_3.

3.1 Cartesian Position/Orientation Vector

The orientation angles r_ρ, r_θ, r_ψ are computed using spherical trigonometry formulas (6) as detailed by Featherstone (1). The equations are the same as equations (13)-(15) in Featherstone except for some typographical errors in equations (13) and (14) that have been corrected here.

From Featherstone we have the following

[13F] $$\cos r_\theta = \cos(\theta_2 + \theta_3)\cos\theta_5 - \sin(\theta_2 + \theta_3)\sin\theta_5\cos\theta_4$$

[14F] $$r_\psi = \theta_6 + \text{atan2}\,[\sin(\theta_2 + \theta_3)\sin\theta_4,\ \sin\theta_5\cos(\theta_2 + \theta_3) + \cos\theta_5\sin(\theta_2 + \theta_3)\cos\theta_4]$$

and

[15F] $$r_\rho = \theta_1 + \text{atan2}\,[\sin\theta_5\sin\theta_4,\ \sin(\theta_2 + \theta_3)\cos\theta_5 + \cos(\theta_2 + \theta_3)\sin\theta_5\cos\theta_4]$$

where atan2 (x, y) is the four-quadrant version of $\tan^{-1}(x/y)$, which is used to avoid problems of angle quadrant ambiguity inherent to trigonometry. As $|\cos r_\theta| \to 1$, the accuracy of equations [14F] and [16F] deteriorates. This is because $|\cos r_\theta| = 1$ is a singular point in the representation of rotations. If $\sin r_\theta = 0$, the orientation of the wrist will depend on the sum of r_ρ and r_ψ and not on their individual values. In this study, if $\sin r_\theta = 0$, r_θ is set to zero or π depending on the sign of $\cos r_\theta$, and the angle r_ψ is set to zero. Equations [11F] and [12F] are then used to solve for r_ρ as follows:

[11F] $\theta_6 = r_\psi - \beta$

but since $r_\psi = 0$

[3.1] $\theta_6 = -\beta$

substituting in [12F]

[12F] $\theta_4 - \beta = \text{atan2}\ \{1/2 \sin (r_\rho - \theta_1)[\cos (\theta_2 + \theta_3) + \cos r_\theta],$

$\cos (r_\rho - \theta_1)\}$

one obtains

[3.2] $r_\rho = \theta_1 + \text{atan2}\ \{2 \sin (\theta_4 + \theta_6)/[\cos (\theta_2 + \theta_3) + \cos r_\theta],$

$\cos (\theta_4 + \theta_6)\}$

To compute the position vector $\underline{r} = (r_x, r_y, r_z)$, the equations for the PUMA 560 are derived next.

Referring to Fig. 3, we can write a set of vector equations that represents the value of r in the three directions x, y, and z as:

[3.3] $\underline{r} = r_x \underline{i} + r_y \underline{j} + r_z \underline{k}$

[3.4] $\underline{r} = \underline{w}_1 + \underline{I}_4$

[3.5] $\underline{w}_1 = \underline{w}_2 + \underline{d}_1 + \underline{\ell}_1$

[3.6] $\underline{w}_2 = \underline{n} + \underline{d}_2$

[3.7] $\underline{n} = \underline{w}_b + \underline{w}_a$

where $\underline{w}_1$ is the vector from the origin of the base coordinate system to the wrist

$\underline{w}_2$ is the vector from the shoulder to the wrist

$\underline{I}_4$ is the vector representing link ℓ_4 and its orientation

$\underline{d}_1$, $\underline{d}_2$ are the vectors representing the offsets d_1 and d_2 and always parallel to the x-y plane

$\underline{\ell}_1$ is the vector representing link ℓ_1 and is in the z direction

$\underline{n}$ vector joining the shoulder to point a

$\underline{w}_a$ projection of $\underline{n}$ on z-axis

$\underline{w}_b$ projection of $\underline{n}$ on x-y plane

There are two alternate ways to solve for $\underline{r}$ as far as the asymmetry existing in link ℓ_3 is concerned. These two methods are presented in the following:

Method 1

Referring again to Fig. 2.3, and substituting θ_3' for $(\theta_3 - \delta)$, the value of the vectors, equations [3.4] to [3.7] can be derived.

First, to find $\underline{n}$, w_a and w_b are expressed as

[3.8] $$w_a = \ell_2 \cos \theta_2 + \ell_3 \cos (\theta_2 + \theta_3')$$

[3.9] $$w_b = \ell_2 \sin \theta_2 + \ell_3 \sin (\theta_2 + \theta_3')$$

Since w_b, the projection of $\underline{n}$ on the x-y plane, makes an angle θ_1 with the y-axis, n, w_2 and w_1 can be written as

[3.10] $$\underline{n} = -w_b \sin \theta_1 \, \underline{i} + w_b \cos \theta_1 \, \underline{j} + w_a \, \underline{k}$$

[3.11] $$\underline{w_2} = (n_x + d_2 \cos \theta_1) \, \underline{i} + (n_y + d_2 \sin \theta_1) \, \underline{j} + w_a \, \underline{k}$$

[3.12] $$\underline{w_1} = [n_x + (d_2 - d_1) \cos \theta_1] \, \underline{i} + [n_y + (d_2 - d_1) \sin \theta_1] \, \underline{j} + (w_a + \ell_1) \, \underline{k}$$

Substituting d for $(d_1 - d_2)$ and from equation [3.10]

[3.13] $$\underline{w_1} = (-w_b \sin \theta_1 - d \cos \theta_1) \, \underline{i} + (w_b \cos \theta_1 - d \sin \theta_1) \, \underline{j} + (w_a + \ell_1) \, \underline{k}$$

The projections of the vector $\underline{I_4}$ on the z-axis and x-y plane are $\ell_4 \cos r_\theta$ and $\ell_4 \sin r_\theta$, respectively. Thus

[3.14] $$\underline{I_4} = -\ell_4 \sin r_\theta \sin r_\rho \, \underline{i} + \ell_4 \sin r_\theta \cos r_\rho \, \underline{j} + \ell_4 \cos r_\theta \, \underline{k}$$

The desired vector $\underline{r}$ can then be obtained by combining equations [3.4], [3.13], and [3.14].

[3.15]
$$r_x = -w_b \sin \theta_1 - d \cos \theta_1 - \ell_4 \sin r_\theta \sin r_\rho$$
$$r_y = w_b \cos \theta_1 - d \sin \theta_1 + \ell_4 \sin r_\theta \cos r_\rho$$
$$r_z = w_a + \ell_1 + \ell_4 \cos r_\theta$$

Method 2

This alternative solution proceeds in the same manner as method 1 except that θ_3 is used in the equations instead of θ_3' which will generate an error E in the wrist position (Fig. 4). Since the angle δ is very small, it follows that

$$E = 2 \sin \delta/2 \, \ell_3 = \ell_3 \delta$$

and also

$$\ell_o = \sin \delta \, \ell_3 = \ell_3 \delta$$

resulting in $E = \ell_o$. To correct for this error, correction terms are added or subtracted from the erroneous position equations to obtain the final result as

$$r_x = - w_b \sin \theta_1 - d \cos \theta_1 - \ell_4 \sin r_\theta \sin r_\rho + \ell_o \sin \theta_1 \cos (\theta_2 + \theta_3 - \delta/2)$$

[3.16]
$$r_y = w_b \cos \theta_1 - d \sin \theta_1 + \ell_4 \sin r_\theta \cos r_\rho - \ell_o \cos \theta_1 \cos (\theta_2 + \theta_3 - \delta/2)$$

$$r_z = w_a + \ell_1 + \ell_4 \cos r_\theta + \ell_o \sin (\theta_2 + \theta_3 - \delta/2)$$

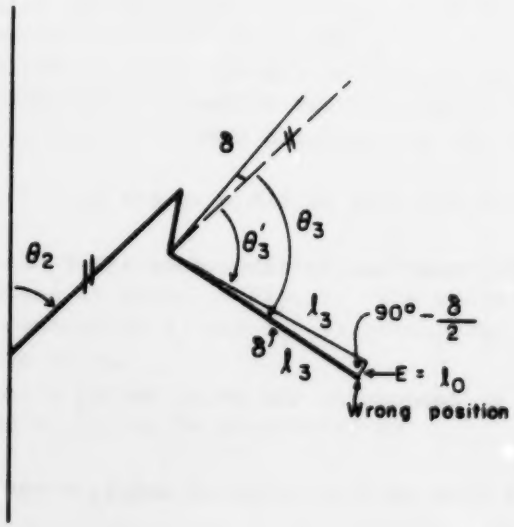


Figure 4. Error in wrist position due to a wrong reference for θ_3 measurement

where ℓ_o is the offset between the centerlines of the two parts constituting the forearm, link ℓ_3, and

$$w_a = \ell_2 \cos \theta_2 + \ell_3 \cos (\theta_2 + \theta_3)$$

$$w_b = \ell_2 \sin \theta_2 + \ell_3 \sin (\theta_2 + \theta_3)$$

3.2 Arm Configuration Parameters

Given a set of joint angles $\underline{\theta}$, the arm configuration is determined by the value of the parameters k_1, k_2, and k_3, which can be evaluated as follows.

3.2.1 Evaluation of k_1

The value of k_1 depends on the arm being right or left. If the arm is lefty, the projection w_b of the arm on the x-y plane is positive, and vice versa for a righty arm. Thus to find k_1 the expression for w_b is calculated as

$$w_b = \ell_2 \sin \theta_2 + \ell_3 \sin (\theta_2 + \theta_3')$$

If $w_b > 0$ then arm is lefty and $k_1 = +1$.

If $w_b < 0$ then arm is righty and $k_1 = -1$.

3.2.2 Evaluation of k_2

The value of k_2 depends on the arm being in the elbow up or elbow down position. Table 1 shows the four different combinations that might exist between k_1 and θ_3. It is concluded that

if $k_1\theta_3 > 0$ then arm is elbow up and $k_2 = +1$

if $k_1\theta_3 < 0$ then arm is elbow down and $k_2 = -1$

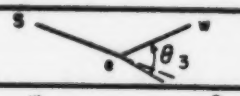
3.2.3 Evaluation of k_3

The value of k_3 depends on the wrist having a no-flip solution or a flip solution.

If $\theta_5 > 0$ then no-flip solution and $k_3 = +1$.

If $\theta_5 < 0$ then flip solution and $k_3 = -1$.

Table 1. Conditions on k_1 and θ_3 for elbow configuration

	K I	θ_3	ELBOW
	+1	>0	UP
	+1	<0	DOWN
	−1	>0	DOWN
	−1	<0	UP

CHAPTER 4. INVERSE KINEMATIC SOLUTION

In this chapter, the inverse kinematics of the PUMA 560 arm are considered. This is the opposite case to the one discussed in the previous chapter, that is, given the Cartesian position/orientation vector $\underline{R} = (r_x, r_y, r_z, r_\rho, r_\theta, r_\psi)$ and the arm configuration parameters k_1, k_2, and k_3, the joint angles vector $\underline{\theta} = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$ is found.

4.1 Computation of θ_1

The angle θ_1 is the result of a rotation of link ℓ_1 around the z-axis. To find θ_1 the position of the wrist w_1 with respect to the base coordinate should be computed using equations [3.4] and [3.14] (see Fig. 3). w_1 is calculated as

$$[4.1] \quad w_1 = (r_x + \ell_4 \sin r_\theta \sin r_\rho)\,\underline{i} + (r_y - \ell_4 \sin r_\theta \cos r_\rho)\,\underline{j} + (r_z - \ell_4 \cos r_\theta)\,\underline{k}$$

To obtain a closed form equation for θ_1 consider the projection ℓ of the arm on the x-y plane as shown in Fig. 5. The projection of ℓ_2 and ℓ_3 on the x-y plane is denoted as ℓ_2' and ℓ_3', respectively. Note that ℓ is equal to the magnitude of w_b.

From Fig. 5b and c, θ_1 can be calculated as

$$[4.2] \quad \theta_1 = 90^\circ - \alpha_1 - k_1\alpha_2$$

where $k_1 = +1$ for a left arm and $k_1 = -1$ for a right arm. The angle α_1 (triangle acd) is found as

$$\alpha_1 = \text{atan2}\ (w_{1y}, -w_{1x})$$

for a left arm and

$$\alpha_1 = \text{atan2}\ (-w_{1y}, w_{1x})$$

for a right arm. Then α_1 can be generally written as

[4.3] $$\alpha_1 = \text{atan2}\ (k_1 w_{1y}, -k_1 w_{1x})$$

Similarly, α_2 (triangle abc) is expressed as

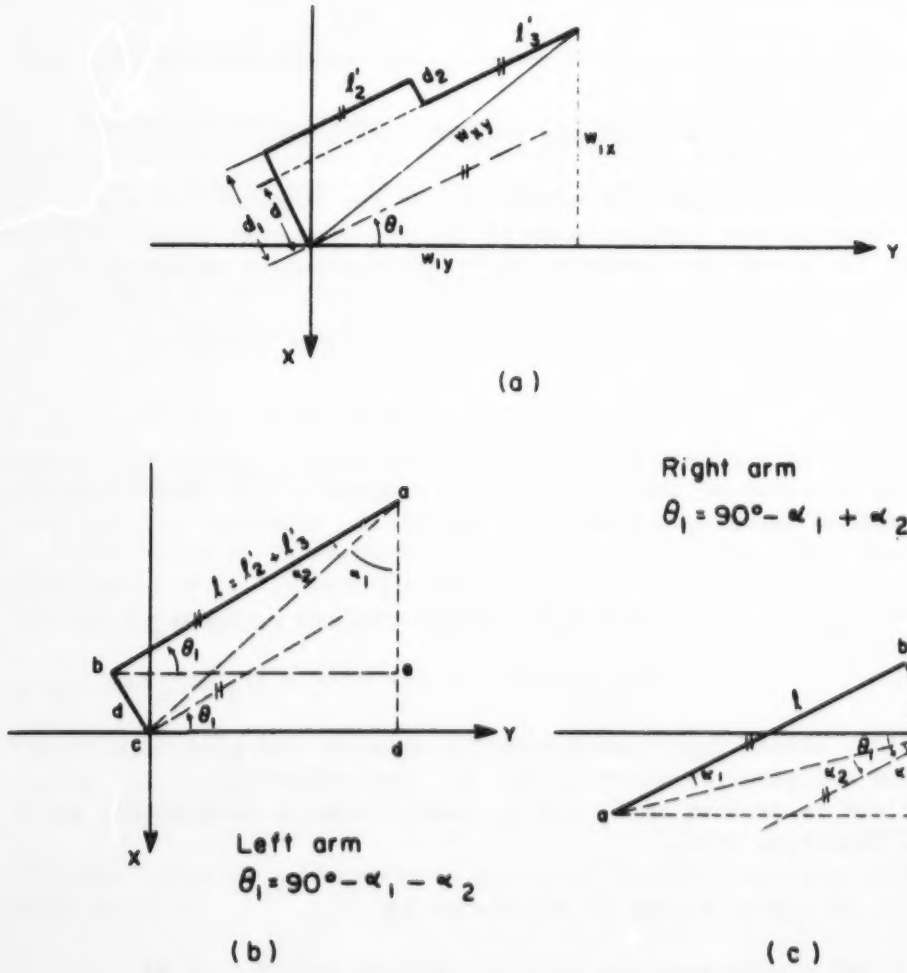


Figure 5. (a) Projection of the first three links on the x, y plane, (b) calculation of θ_1, for left arm, (c) calculation of θ_1, for a right arm

[4.4] $\alpha_2 = \text{atan2}\ (d, \ell);\quad 0 \leq \alpha_2 \leq 90°$

where

$$\ell = (w_{xy}^2 - d^2)^{1/2}$$

Thus, θ_1 can be expressed as

[4.5] $\theta_1 = \text{atan2}\ (-k_1 w_{1x}, k_1 w_{1y}) - k_1\ \text{atan2}\ (d, \ell)$

Equation [4.5] indicates that a singular point exists if $w_{1x} = w_{1y} = 0$. Considering the arm geometry, this condition is never satisfied. The point where both w_{1x} and w_{1y} are at their absolute minimum value is obtained when the wrist is pointing up such that points a and b coincide. In this case, for the left arm for example, $w_{1x} = -d \cos \theta_1$, $w_{1y} = -d \sin \theta_1$, and $\ell = 0$, resulting in an $\alpha_2 = 90°$ and a $\theta_1 = -\alpha_1$.

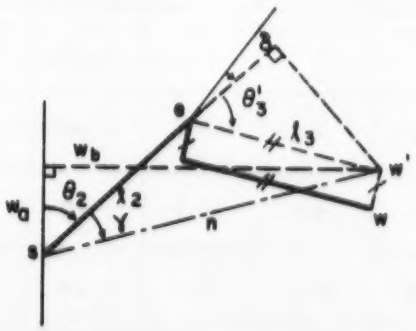
4.2 Computation of θ_2 and θ_3

To find the joint angles θ_2 and θ_3, consider Fig. 6, which is a representation of links ℓ_2 and ℓ_3 and the offset d_2. Also shown on the figure are the different angles used in the computation of θ_2 and θ_3.

From Fig. 6 and by the application of the cosine rule to triangle sew', one obtains

[4.6] $$\cos \theta_3' = \frac{n^2 - \ell_2^2 - \ell_3^2}{2\ell_2\ell_3}$$

If $|\cos \theta_3'| > 1$ then the position is unobtainable and if $|\cos\theta_3'| = 1$, the manipulator is at a dead point.



$$W_b^2 = n_x^2 + n_y^2$$

$$W_a = W_{1z} - \ell_1$$

Figure 6. Calculation of θ_2 and θ_3 for a left arm

Table 2 gives the different possibilities of the arm configuration for the first three links and the sign of the different angles. From Table 2, it is noticed that θ_3' and γ have the same sign, which corresponds also to the product k_1k_2. Hence we can deduce that θ_3 for any arm configuration is represented by the following equation

$$[4.7] \qquad \theta_3 = k_1 \, k_2 \, \theta_3' + \delta$$

where δ is as defined in equation [2.2].

Also, from Fig. 6 and Table 2, the following relations are obtained

$$[4.8] \qquad \begin{aligned} \theta_2 + \gamma &= \text{atan2}\,(w_b, w_a) \\ \theta_2 + \gamma &= k_1 \, \text{atan2}\,(\ell, w_{1z} - \ell_1) \end{aligned}$$

and

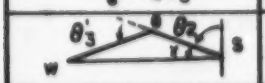
$$[4.9] \qquad \tan \gamma = \ell_3 \sin \theta_3' / (\ell_2 + \ell_3 \cos \theta_3')$$

or

$$[4.10] \qquad \gamma = k_1k_2 \, \text{atan2}\,(\ell_3 \sin \theta_3', \ell_2 + \ell_3 \cos \theta_3')$$

Thus, θ_2 can be computed from equations [4.8] and [4.10] as

Table 2. Arm configuration for the first three links and the sign of the angles involved

	K_1	K_2	K_1K_2	θ_2	θ_3'	γ
	+ 1	+ 1	+ 1	+ ve	+ ve	+ ve
	+ 1	− 1	− 1	+ ve	− ve	− ve
	− 1	+ 1	− 1	− ve	− ve	− ve
	− 1	− 1	+ 1	− ve	+ ve	+ ve

[4.11] $\theta_2 = k_1[\text{atan2}\ (\ell_2, w_{1x} - \ell_1)$

$$- k_2\ \text{atan2}\ (\ell_3 \sin\theta_3', \ell_2 + \ell_3 \cos\theta_3')]$$

4.3 Computation of θ_4, θ_5, and θ_6

To find the angles θ_4, θ_5, and θ_6, Featherstone equations can be applied directly.

First, θ_5 is calculated from the following

[8F] $$\cos\theta_5 = \cos(\theta_2 + \theta_3)\cos r_\theta + \sin(\theta_2 + \theta_3)\sin r_\theta \cos(r_\rho - \theta_1)$$

If $|\cos\theta_5| \neq 1$, $\sin\theta_5$ is computed as $(1 - \cos^2\theta_5)^{1/2}$. If $\sin\theta_5 > \varepsilon$, where ε is a small number, θ_4 and θ_6 can be found from equations [9F] to [11F] as

[9F] $\theta_4 = \text{atan2}\ [\sin r_\theta \sin(r_\rho - \theta_1), \cos(\theta_2 + \theta_3)\sin r_\theta \cos(r_\rho - \theta_1)$

$$- \sin(\theta_2 + \theta_3)\cos r_\theta]$$

and

[11F] $$\theta_6 = r_\psi - \beta$$

where

[10F] $\beta = \text{atan2}\ [\sin(\theta_2 + \theta_3)\sin(r_\rho - \theta_1), \sin r_\theta \cos(\theta_2 + \theta_3)$

$$- \cos r_\theta \sin(\theta_2 + \theta_3)\cos(r_\rho - \theta_1)]$$

The accuracy of equations [9F] and [10F] deteriorates as $\sin\theta_5 \to 0$ and they break down completely if $\sin\theta_5 = 0$. Therefore, for some value of ε and for $\sin\theta_5 < \varepsilon$, better correspondence between θ_4 and β can be obtained by using the expression

[12F] $\theta_4 - \beta = \text{atan2}\ \{(1/2)\sin(r_\rho - \theta_1)[\cos(\theta_2 + \theta_3) + \cos r_\theta),$

$$\cos(r_\rho - \theta_1)]\}$$

in place of either [9F] or [10F].

In the case of a no-flip condition, that is $k_3 = 1$, the wrist angles are obtained from the above equations. If, however, $k_3 = -1$, the flip solution becomes $(\theta_4 + \Pi, -\theta_5, \theta_6 + \Pi)$

CHAPTER 5. SUMMARY AND DISCUSSION

Efficient algorithms for the calculation of the direct and inverse kinematics for the PUMA 560 robot have been presented. These algorithms are based on a method (1), that takes direct advantage of the geometry of the manipulator.

The transformation from joint angles to Cartesian coordinates executes slightly faster than those in the opposite direction, see Table 3. The comparison of the complexity of the present inverse kinematic solution with Paul et al. (4) proves the present solution more efficient.

To be able to compare the two methods, link ℓ_4 was set to zero (as in reference 4), that is, w_{1x}, w_{1y}, and w_{1z} of chapters 3 and 4 of this report are equivalent to p_x, p_y, and p_z in reference 4. Also, for comparison, multiplications due to arm configuration are not considered since Paul et al. (4) did not consider them and the orientation is assumed to be given in Euler angles. A difference between both solutions is in the way the orientation is expressed: in Paul et al. the orientation is expressed as the three vectors $\underline{n}$, $\underline{o}$, and $\underline{a}$, while in this report it is presented as the three rotations r_ρ, r_θ, and r_ψ. The cost to go from $\underline{n}$, $\underline{o}$, $\underline{a}$ to Euler angles is 9 multiplies, 3 additions, and 5 transcendental functions. The reverse way is more expensive: 14 multiplies, 4 additions, and 6 transcendental functions. To go from r_ρ, r_θ, r_ψ to Euler angles or vice-versa it takes only 2 additions.

The real advantage of the present method is in the efficiency of the calculation of the velocity and acceleration transformations. In order to find the kinematic velocities (accelerations), one uses intermediate and final results of the kinematic positions (velocities). Since for the

Table 3. Computation complexity for the kinematics solution of the PUMA 560 arm

	DIRECT			INVERSE		
	a	m	f	a	m	f
Paul's approach	25	53	17	32	70	22
Present approach	21	25	17	21	31	16

efficient control of the arm all these solutions are needed, there is quite a saving in computation time. The development of equations and solutions for the kinematic velocities and accelerations will be the subject of a succeeding report.

REFERENCES

1. Featherstone, R. Position and velocity transformations between robot end-effector coordinates and joint angles. Int. J. Robot. Res. 2(2): 35-45; 1983.

2. Lee, C.S.G. Robot arm kinematics, dynamics and control. IEEE Trans. Comput. 15(12): 62-80; 1982.

3. Paul, R.P. Robot Manipulators: Mathematics, Programming, and Control. Cambridge, Mass. MIT Press 1981.

4. Paul, R.P., Shimano, B., and Meyer, G.E. Kinematic control equations for simple manipulators. IEEE Trans. Syst., Man Cybern. SMC-11(6): 449-455; 1981.

5. Uchiyama, M. A study of computer control of motion of a mechanical arm. Bull. Jpn. Soc. Mech. Eng. 22(173): 1640-1647. 1979.

6. Duffy, J. Analysis of Mechanisms and Robot Manipulators. London: Edward Arnold. 1980.

APPENDIX A. PUMA 560 PARAMETERS

A1. LINK CONSTANTS

ℓ_1 = 672 mm, length of link 1, from base to shoulder

ℓ_2 = 432 mm, length of link 2, upper arm

ℓ_3 = 433 mm, length of link 3, lower arm

ℓ_4 = 56 mm, length of link 4, from center of wrist to flange

d = 149.1 mm, net offset in the arm due to shoulder offset d_1 and elbow offset d_2

ℓ_o = 20.5 mm, offset between center lines of the two parts of the forearm, link 3.

A2. TRANSFORMATIONS

A2.1 Transformation Between θ_{PUMA}/R_{PUMA} and θ/R as Defined in this Report.

θ_{PUMA} vs θ:

[A1]

$$\theta_{1p} = \theta_1 + 90 \qquad \theta_{4p} = \theta_4$$
$$\theta_{2p} = \theta_2 - 90 \qquad \theta_{5p} = \theta_5$$
$$\theta_{3p} = \theta_3 + 90 \qquad \theta_{6p} = \theta_6 - 90$$

R_{PUMA} vs R:

[A2]

$$x_p = r_x \qquad O = r_p + 180$$
$$y_p = r_y \qquad A = r_\theta - 90$$
$$z_p = r_z - \ell_1 \qquad T = r_\psi - 90$$

A2.2 Transformation between Euler Angles (ϕ_e, θ_e, ψ_e) and PUMA's (O, A, T) Angles

[A3]

$$\phi_e = O - 90$$
$$\theta_e = A + 90$$
$$\psi_e = T$$

A3. JOINT LIMITS

The optimum way is to check every joint angle for limits immediately after its computation. The ranges for the joint angles are as follows:

[A4]

$$-250 < \theta_1 < 70 \qquad -110 < \theta_4 < 170$$
$$-133 < \theta_2 < 133 \qquad -100 < \theta_5 < 100$$
$$-142 < \theta_3 < 142 \qquad -176 < \theta_6 < 356$$

If θ_4 and/or θ_6 hit a joint limit, the wrist configuration (no-flip/flip) changes so that these angles stay within the limits.

The range on joint angle 6 is more than 360°, see Fig. A1. In the program, the solution for θ_6 always gives a value within the 360° in the range [-90, 270]. If θ_6 falls in the ranges [-90, -4] or [184, 270], it might be off by 360° and should be in the ranges [270, 356] or [-176, -90] respectively. Thus one should keep track, while moving, of the rotation of θ_6 and add or subtract 360° to θ_6 as needed.

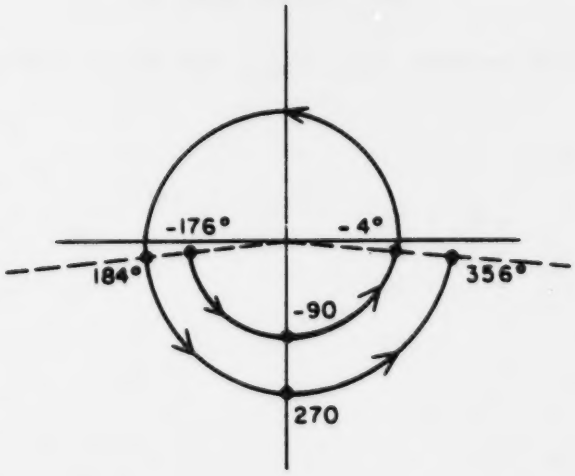


Figure A1. Range on joint angle θ_6

APPENDIX B. PROGRAM LISTINGS

The listings of the programs written for the kinematic positions are presented in this appendix.

The C-function 'jt-cart' solves for the direct kinematics while the function 'cart-jt' solves for the inverse kinematics. The main program 'mainkp.c' reads first the type of transformation to be performed. In the case of a direct transformation main accepts the joint angles either as referenced by Unimation and transform them to the format defined in this report using equation [A1] or accepts them directly as defined here.

In the case of the inverse kinematics, main accepts the Cartesian coordinates either in the PUMA format (x_p, y_p, z_p, O, A, T) or using for the orientation of the arm, either the Euler angles (ϕ_e, θ_e, ψ_e) or the rotation angles (r_ρ, r_θ, r_ψ) as defined in this report. The main program uses equations [A2] and [A3] to transform the orientation to its equivalent (r_ρ, r_θ, r_ψ). The solution is then outputted in the same input format.

```
/*   mainkp.c   */


#include <std.h>
#include "defin.h"
#include "atan.c"
#include "kinem.c"

extern float rx=0,ry=0,rz=0,rho=0,rtheta=0,reps=0;
extern float th1=0,th2=0,th3=0,th4=0,th5=0,th6=0;
extern int status=0,k1=1,k2=1,k3=1;


main()

    /*  Reads either Cartesian coord. of a point (translation in
        millimetres and rotations in degrees) or joint angles (in degrees)
        and does the necessary transformations.

          The input variables are:
            Euler = 1 ---> orientation expressed in Euler angles and
                           joint angles referenced as in report
                  = 0      otherwise
             oat  = 1 ---> orientation expressed in O, A, T angles and
                           joint angles referenced to the PUMA
                  = 0      otherwise
         jt_to_cart = 1 ---> transform is from jt angles to cart. coord.
                    = 0    transform is from cart. coord. to jt angles */


{
    extern float rx,ry,rz,rho,rtheta,reps;
    extern float th1,th2,th3,th4,th5,th6;
    extern int status,k1,k2,k3;
    int oat,euler,jt_to_cart;
    float o,a,t,ephi,etheta,eeps;

    putfmt (" euler = \n");
    getfmt ("%i",&euler);
    putfmt (" oat = \n");
    getfmt ("%i",&oat);
    putfmt (" jt_to_cart = \n");
    getfmt ("%i",&jt_to_cart);

    if ( jt_to_cart == 1 )
    {
```

```
putfmt (" th1 to th6 ; one per line (8.3)\n");
getfmt ("%8.3f",&th1);
getfmt ("%8.3f",&th2);
getfmt ("%8.3f",&th3);
getfmt ("%8.3f",&th4);
getfmt ("%8.3f",&th5);
getfmt ("%8.3f",&th6);
putfmt ("    th1       th2        th3         th4        th5");
putfmt ("         th6\n");
putfmt ("%9.3f %9.3f %9.3f %9.3f %9.3f %9.3f\n",
        th1,th2,th3,th4,th5,th6);
if ( oat == 1)
{
        th1 = th1 - 90;
        if ( th1 > 0 && th1 > MAX1 && th1 < MIN1 )
        {
                putfmt (" limit on jt1 \n");
                exit();
        }
        th2 = th2 + 90;
        if ( abs(th2) > MAX2 && abs(th2) < 180 )
        {
                putfmt (" limit on jt2 \n");
                exit();
        }
        th3 = th3 - 90;
        if ( abs(th3) > MAX3 && abs(th3) < 180 )
        {
                putfmt (" limit on jt3 \n");
                exit();
        }
        th6 = th6 + 90;
if ((th4 > MAX4 && th4 < 180) || (th4 < MIN4 && th4 > -180))
        {
                putfmt (" limit on jt4 \n");
                exit();
        }
        if ( abs(th5) > MAX5 && abs(th5) < 180 )
        {
                putfmt (" limit on jt5 \n");
                exit();
        }
        if ( th6 > MAX6 || th6 < MIN6 )
        {
                putfmt (" limit on jt6 \n");
                exit();
        }
```

```
    }
    jt_cart ();
    putfmt (" k1 =%i k2 =%i k3 =%i\n",k1,k2,k3);
    if ( euler == 1 )
    {
            ephi = rho + 90;
            if (ephi > 180 ) ephi = ephi - 360;
            if (ephi < -180) ephi = ephi + 360;
            etheta = rtheta;
            eeps = reps - 90;
            if ( eeps > 180 ) eeps = eeps - 360 ;
            if ( eeps < -180) eeps = eeps + 360 ;
            putfmt("    rx       ry       rz      ephi    etheta");
            putfmt ("    eeps\n");
            putfmt ("%9.3f %9.3f %9.3f %9.3f %9.3f %9.3f\n",
                    rx,ry,rz,ephi,etheta,eeps);
    }
    else if ( oat == 1 )
    {
            o = rho + 180;
            a = rtheta - 90;
            t = reps -90;
            if ( o > 180 ) o = o - 360 ;
            if ( o < -180 ) o = o + 360 ;
            if ( a > 180 ) a = a - 360 ;
            if ( a < -180 ) a = a + 360 ;
            if ( t > 180 ) t = t - 360 ;
            if ( t < -180 ) t = t + 360 ;
            rz = rz - L1;
            putfmt("    rx       ry       rz        o         a");
            putfmt ("        t\n");
            putfmt ("%9.3f %9.3f %9.3f %9.3f %9.3f %9.3f\n",
                    rx,ry,rz,o,a,t);
    }
    else
    {
            putfmt("    rx       ry       rz      rho     rtheta");
            putfmt ("    reps\n");
            putfmt ("%9.3f %9.3f %9.3f %9.3f %9.3f %9.3f\n",
                    rx,ry,rz,rho,rtheta,reps);
    }
}
else {
    putfmt (" k1,k2,k3 ; one per line \n");
    getfmt ("%i",&k1);
    getfmt ("%i",&k2);
    getfmt ("%i",&k3);
```

```
if (euler == 1)
{
        putfmt ("rx,ry,rz,ephi,etheta,eeps;one per line(8.3)\n");
         getfmt ("%8.3f",&rx);
         getfmt ("%8.3f",&ry);
         getfmt ("%8.3f",&rz);
         getfmt ("%8.3f",&ephi);
         getfmt ("%8.3f",&etheta);
         getfmt ("%8.3f",&eeps);
         putfmt("    rx       ry       rz      ephi    etheta");
         putfmt ("    eeps\n");
         putfmt ("%9.3f %9.3f %9.3f %9.3f %9.3f %9.3f\n",
                 rx,ry,rz,ephi,etheta,eeps);
         rho = ephi - 90;
         rtheta = etheta ;
         reps = eeps + 90;
         cart_jt();
}
else if ( oat == 1 )
{
       putfmt ("rx,ry,rz,o,a,t; one per line(8.3)\n");
         getfmt ("%8.3f",&rx);
         getfmt ("%8.3f",&ry);
         getfmt ("%8.3f",&rz);
         getfmt ("%8.3f",&o);
         getfmt ("%8.3f",&a);
         getfmt ("%8.3f",&t);
         putfmt("    rx       ry       rz        o          a");
         putfmt ("       t\n");
         putfmt ("%9.3f %9.3f %9.3f %9.3f %9.3f %9.3f\n",
                 rx,ry,rz,o,a,t);
         rz = rz + L1 ;
         rho = o - 180 ;
         rtheta = a + 90 ;
         reps = t + 90 ;
         cart_jt ();
}

else
{
       putfmt ("rx,ry,rz,rho,rtheta,reps; one per line(8.3)\n");
         getfmt ("%8.3f",&rx);
         getfmt ("%8.3f",&ry);
         getfmt ("%8.3f",&rz);
         getfmt ("%8.3f",&rho);
         getfmt ("%8.3f",&rtheta);
         getfmt ("%8.3f",&reps);
```

```
        putfmt("    rx        ry        rz       rho      rtheta");
        putfmt ("    reps\n");
        putfmt ("%9.3f %9.3f %9.3f %9.3f %9.3f %9.3f\n",
                rx,ry,rz,rho,rtheta,reps);
        cart_jt ();
}

switch (status)
{
        case -1:
                putfmt (" position unobtainable \n");
                break;
        case 1:
                putfmt (" limit on joint 1 \n");
                break;
        case 2:
                putfmt (" limit on joint 2 \n");
                break;
        case 3:
                putfmt (" limit on joint 3 \n");
                break;
        case 4:
                putfmt (" limit on joint 4 \n");
                break;
        case 5:
                putfmt (" limit on joint 5 \n");
                break;
        case 6:
                putfmt (" limit on joint 6 \n");
                break;
        default:
                if ( oat == 1 )
                {
                        th1 = th1 + 90 ;
                        th2 = th2 - 90 ;
                        th3 = th3 + 90 ;
                        th6 = th6 - 90 ;
                        if ( th1 > 180 ) th1 = th1 - 360 ;
                        if ( th1 < -180 ) th1 = th1 + 360 ;
                        if ( th2 > 180 ) th2 = th2 - 360 ;
                        if ( th2 < -180 ) th2 = th2 + 360 ;
                        if ( th3 > 180 ) th3 = th3 - 360 ;
                        if ( th3 < -180 ) th3 = th3 + 360 ;
                }
```

```
                putfmt ("    th1       th2       th3       th4       th5");
                putfmt ("      th6\n");
                putfmt ("%9.3f %9.3f %9.3f %9.3f %9.3f %9.3f\n",
                        th1,th2,th3,th4,th5,th6);
        }
    }
}
```

```
/*   defin.h   */

#define   L1    672
#define   L2    432
#define   L3    433
#define   L4     56
#define   D     149.1
#define   AL     20.5
#define   ANG    2.72
#define   EPS    0.05
#define   RAD    0.0174533
#define   DEG    57.2957795
#define   MAX1   70
#define   MIN1   110
#define   MAX2   133
#define   MAX3   142
#define   MAX4   170
#define   MIN4   -110
#define   MAX5   100
#define   MAX6   356
#define   MIN6   -176
```

```
                        /* atan.c */


float atan2 ( m,n )          /* computes right quadrant for */
                            /*   the arc tangent function    */
float m,n;
{
        float ang;
        double arctan();

        if ( m == 0 && n == 0 )
        {
                putfmt (" INVALID PARAMETERS for atan2(m,n) !!\n");
                exit();
        }
        if ( n==0 ) return ( (m<0) ? -90.0 : 90.0 );
        if ( m==0 ) return ( (n<0) ? 180.0 :  0.0 );

        ang = DEG * (arctan((double) (abs(m) / abs(n)) ));

        if ( m>0 && n<0 ) return ( 180. - ang );
        if ( m<0 && n<0 ) return ( ang - 180. );
        if ( m<0 && n>0 ) return ( - ang );
        return ( ang);
}
```

/* kinem.c */

```
jt_cart ( )
                        /* transforms joint coord. to cartesian */
                              /* coord. ( method 1 )     */
{
    extern float th1,th2,th3,th4,th5,th6;
    extern float rx,ry,rz,rho,rtheta,reps;
    extern int k1,k2,k3;
    double sin(),cos(),sqrt();
    double t1,t2,t3,t4,t5,t6,t23,t46;
    float  atan2(),c46,s46,rxw,ryw,rzw;
    float  c1,s1,c2,s2,c23,s23,c4,s4,c5,s5,crtheta,srtheta;
    float  a1,a2,num1,num2,den1,den2,rh,wa,wb,cr,sr,a,rr,num;

    t1 = th1 * RAD;
    t2 = th2 * RAD;
    t3 = th3 * RAD;
    t4 = th4 * RAD;
    t5 = th5 * RAD;
    t6 = th6 * RAD;
    t23 = t2 + t3 ;
    t46 = t4 + t6 ;

    c1 = cos ( t1 );
    s1 = sin ( t1 );
    c2 = cos ( t2 );
    s2 = sin ( t2 );
    c23 = cos ( t23 );
    s23 = sin ( t23 );
    c4 = cos ( t4 );
    s4 = sin ( t4 );
    c5 = cos ( t5 );
    s5 = sin ( t5 );

    crtheta = c23 * c5 - s23 * s5 * c4;
    if ( abs( crtheta ) != 1 )
    {
         a1 = s5 * c23;
         a2 = c5 * s23;
         num1 = s23 * s4;
         num2 = s5 * s4;
         den1 = a1 + a2 * c4;
         den2 = a2 + a1 * c4;
         reps = th6 + atan2 ( num1 , den1 );
```

```
        rho = th1 + atan2 ( num2 , den2 );
        srtheta = sqrt ((double) (1 - crtheta * crtheta ));
        rtheta = atan2 ( srtheta, crtheta );
    }
    else
    {
        c46 = cos ( t46 );
        s46 = sin ( t46 );
        if ( crtheta == 1) rtheta = 0;
        else rtheta = 180;
        srtheta = 0;
        reps = 0;
        num = ( 2 * s46 ) / ( c23 + crtheta );
        rho = th1 + atan2 ( num , c46 );
    }


    a = L4 * srtheta;
    rr = rho * RAD;
    sr = sin ((double) (rr));
    cr = cos ((double) (rr));
    t23 = ANG * RAD ;
    c23 = cos ( t23 );
    s23 = sin ( t23 );

    wa = c2 * L2 + c23 * L3;
    wb = s2 * L2 + s23 * L3;

    rxw = -s1 * wb - D * c1 ;
    ryw = c1 * wb - D * s1 ;
    rzw = wa + L1 ;

    rx = rxw - a * sr ;
    ry = ryw + a * cr ;
    rz = rzw + L4 * crtheta ;

    /* arm configuration */
    if ( wb < 0 ) k1 = -1;
    if ( k1*th3 < 0 ) k2 = -1;
    if ( th5 < 0 ) k3 = -1;
}
```

```
cart_jt ()                  /* transforms cartesian  coord. to */
                            /* joint angles                    */

{
     extern float th1,th2,th3,th4,th5,th6;
     extern float rx,ry,rz,rho,rtheta,reps;
     extern int status,k1,k2,k3;
     double cos(),sin(),sqrt();
     float atan2();
     float w1[3],wa,n2;
     float num1,den1,c3p,s3p,a2,b2;
     float c23,s23,cr1,sr1,c5,s5,c4,s4,s4b;
     float ctheta,stheta,crho,srho,cbeta,sbeta,beta;
     float rth,rh,a,t23,t1,len;

     rth = rtheta * RAD;
     rh = rho * RAD;
     ctheta = cos (rth);
     stheta = sin (rth);
     crho = cos (rh);
     srho = sin (rh);

     a = L4 * stheta;
     w1[0] = rx + a * srho ;
     w1[1] = ry - a * crho ;
     w1[2] = rz - L4 * ctheta ;

     len = (double) ( w1[0] * w1[0] + w1[1] * w1[1] - D*D );
     if ( len < 0 )   return ( status = -1 );
     len = sqrt ( len );

     if ( k1 > 0 ) th1 = atan2 ( - w1[0] , w1[1] ) - atan2 (D,len);
     if ( k1 < 0 ) th1 = atan2 ( w1[0] , -w1[1] ) + atan2 (D,len);
     if ( th1 > 180 ) th1 = th1 - 360 ;

     /* limit check on th1 */
     if ( th1 > 0 && th1 > MAX1 && th1 < MIN1 ) return (status = 1);

     t1 = th1 * RAD ;
     wa = w1[2] - L1 ;
     n2 = len * len + wa * wa ;

     c3p = ( n2 - L2 * L2 - L3 * L3 ) / ( 2 * L2 * L3 );
     if ( abs(c3p) > 1.001 ) return (status= -1);
                              /* causes main to stop the arm */
                              /*and to give following message*/
                              /* " position unobtainable "   */
```

```
if ( abs(c3p) <= 1.001 && abs(c3p) > 1 ) c3p = c3p /abs(c3p);
if (abs(c3p) == 1) putfmt ("WARNING.. maximum reach — jt3 \n");
s3p = sqrt ((double) ( 1 - c3p * c3p ));
th3 = k1 * k2 * atan2 (s3p,c3p) + ANG ;

/* limit check for th3 */
if ( abs(th3) > MAX3 && abs(th3) < 180 ) return ( status = 3);

a2 = L3 * s3p;
b2 = L2 + L3 * c3p;
th2 = k1 * ( atan2( len,wa ) - k2 * atan2 (a2,b2)) ;

/* limit check for th2 */
if ( abs(th2) > MAX2 && abs(th2) < 180 ) return ( status = 2 );

t23 = RAD * ( th2 + th3 );
c23 = cos ((double) (t23));
s23 = sin ((double) (t23));
cr1 = cos ((double) ( rh - t1 ));
sr1 = sin ((double) ( rh - t1 ));
c5 = c23 * ctheta + s23 * stheta * cr1;
s5 = sqrt ((double) ( 1 - c5 * c5 ));
th5 = atan2 ( s5, c5);
if ( s5 > 0.001) {
     sbeta = s23 * sr1;
     cbeta = stheta * c23 - ctheta * s23 *cr1;
     beta = atan2 (sbeta, cbeta );
     th6 = reps - beta ;

     if ( s5 > EPS )
     {
             s4 = stheta * sr1;
             c4 = c23 * stheta *cr1 - s23 * ctheta ;
             th4 = atan2 ( s4, c4 );
     }
     else
     {
             s4b = 0.5 * sr1 * ( c23 + ctheta );
             th4 = atan2 ( s4b, cr1) + beta ;
     }
}
else                              /* straight wrist */
{
     putfmt ("WARNING .. straight wrist \n");
     beta = 0 ;
     th6 = reps ;
```

```
        s4b = 0.5 * sr1 * ( c23 + ctheta );
        th4 = atan2 ( s4b, cr1) + beta ;
    }

    /* limit check for th4, th5, th6 */
    if ( abs(th5) > MAX5 && abs(th5) < 180 ) return ( status = 5 );
    if (((th4 > MAX4 && th4 < 180 ) || (th4 < MIN4 && th4 > -180 ))
       || ( th6 > MAX6 || th6 < MIN6 ) || ( k3 == -1 ))
    {
        th5 = - th5;
        th4 = th4 + 180;
        if ( th4 > 180 ) th4 = th4 - 360;
        if ( th4 < -180 ) th4 = th4 + 360;
        th6 = th6 + 180;
        if ( th6 > 360 ) th6 = th6 - 360;
    }
    if (( th4 > MAX4 && th4 < 180 ) || ( th4 < MIN4 && th4 > -180 )
        return ( status = 4 );
    if ( th6 > MAX6 || th6 < MIN6 ) return ( status = 6 );
}
```